Count Data Regression: Poisson GLMs

STAT 245

School Survey on Crime

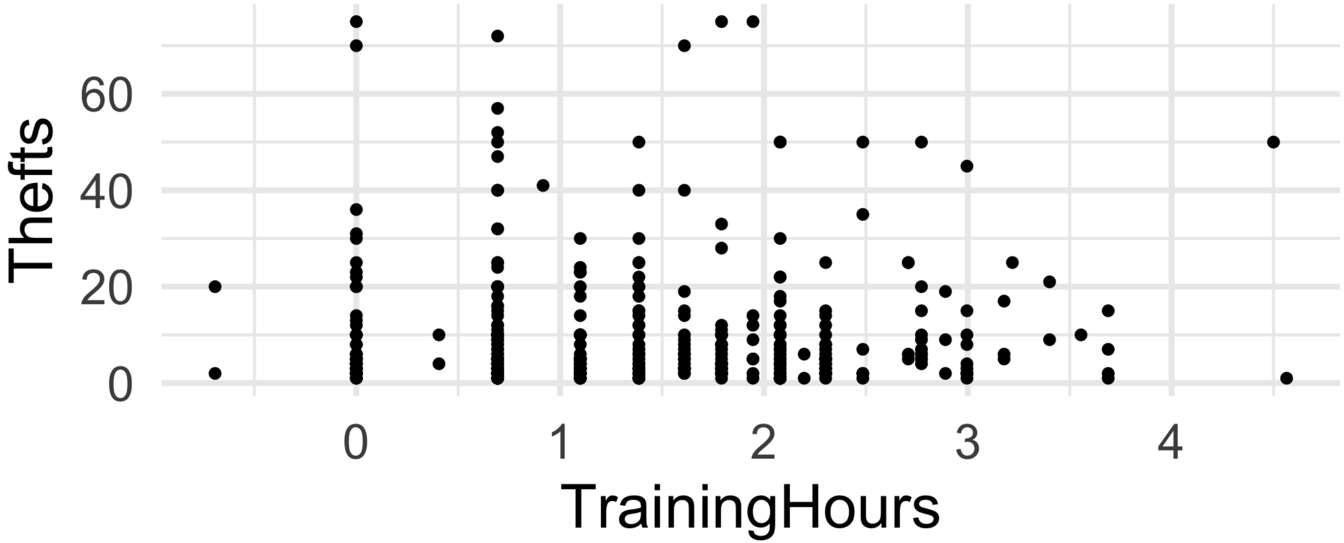
Today's dataset was collected by surveying school administrators across the US about crimes and violent incidents that took place in their school (as well as some characteristics of each school). We will try to fit a model to predict the number of **thefts** reported at each school.

Plan: Candidate Predictors

- `Security`
- `SecurityCameras`
- `Lockers`
- `LockedGates`
- `Location` (City, Town, Urban Fringe, Rural)
- `NEnrollment` (number of students)
- `TrainingHours` for staff on prevention

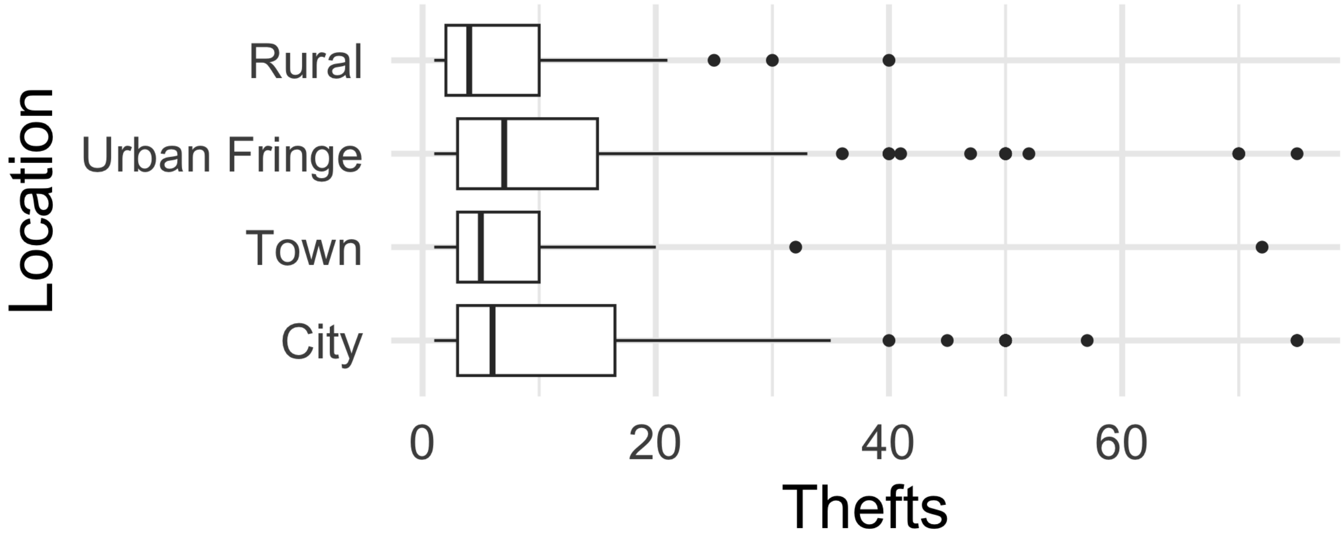
Graphs

```
gf_point(Thefts ~ TrainingHours, data = sscrime)
```



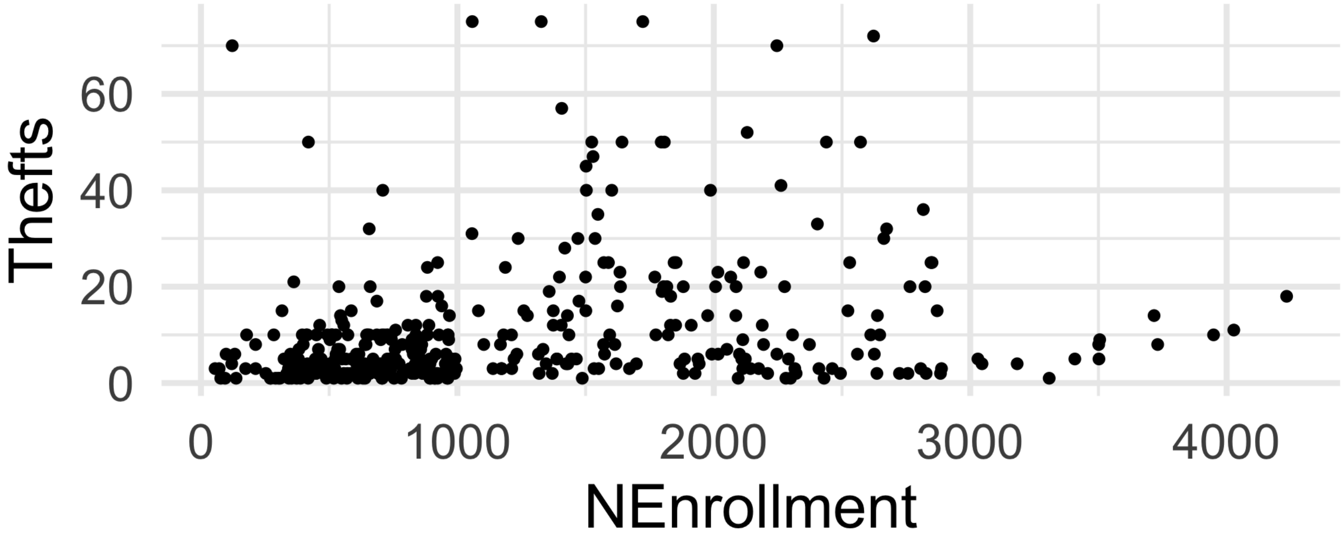
Graphs

```
gf_boxplot(Location ~ Thefts,
           data = sscrime)
```



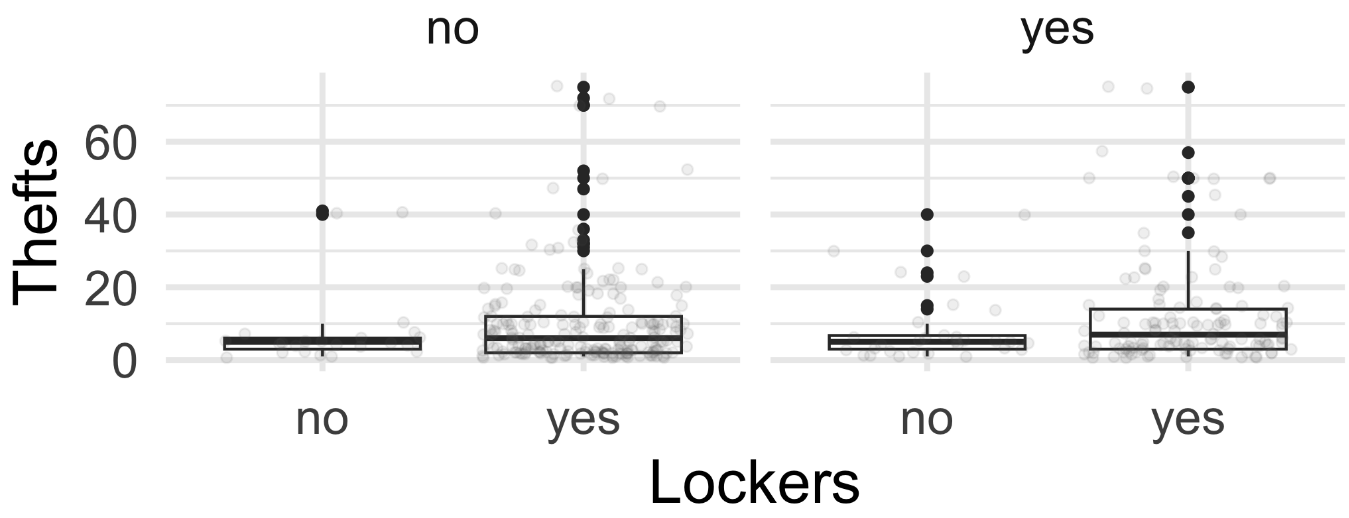
Graphs

```
gf_point(Thefts ~ NEnrollment, data = sscrime)
```



Graphs

```
gf_boxplot(Thefts ~ Lockers | LockedGates,
           data = sscrime) |>
  gf_jitter(color = 'grey44', alpha = 0.1)
```



Multiple Linear Regression

This is NOT going to go well...

doing it to show why we need a new kind of model

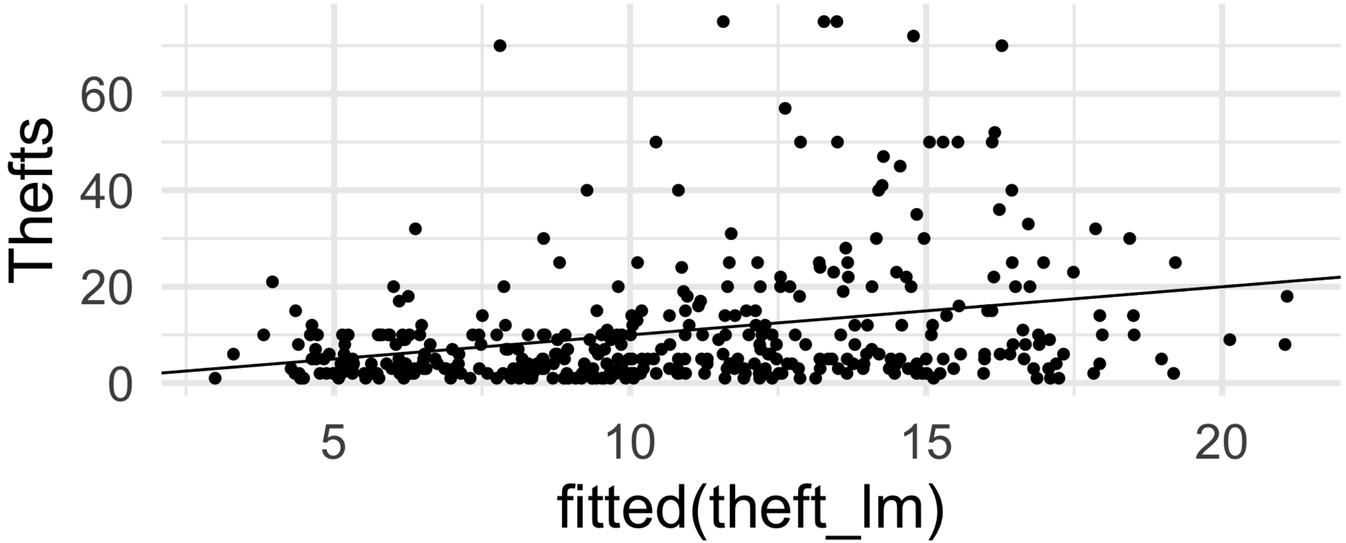
```r
theft_lm <- lm(Thefts ~ NEnrollment + Location +
                 TrainingHours + SecurityCameras,
               data = sscrime)
```

```r
summary(theft_lm)
```

```
##
## Call:
## lm(formula = Thefts ~ NEnrollment + Location + TrainingHours +
##     SecurityCameras, data = sscrime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.181  -7.720  -3.109   3.038  63.427
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             8.398596   2.074867   4.048 6.29e-05 ***
## NEnrollment             0.003130   0.000829   3.775 0.000186 ***
## LocationTown           -3.894781   2.213527  -1.760 0.079304 .
## LocationUrban Fringe -0.978436   1.682101  -0.582 0.561136
## LocationRural          -4.670431   1.908254  -2.447 0.014845 *
## TrainingHours          -0.265514   0.721441  -0.368 0.713057
## SecurityCamerasyes      2.257084   1.395293   1.618 0.106583
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.8 on 374 degrees of freedom
## Multiple R-squared:  0.08738,    Adjusted R-squared:  0.07274
## F-statistic: 5.968 on 6 and 374 DF,  p-value: 5.684e-06
```

Response vs Fitted: *optional* way to see goodness of fit

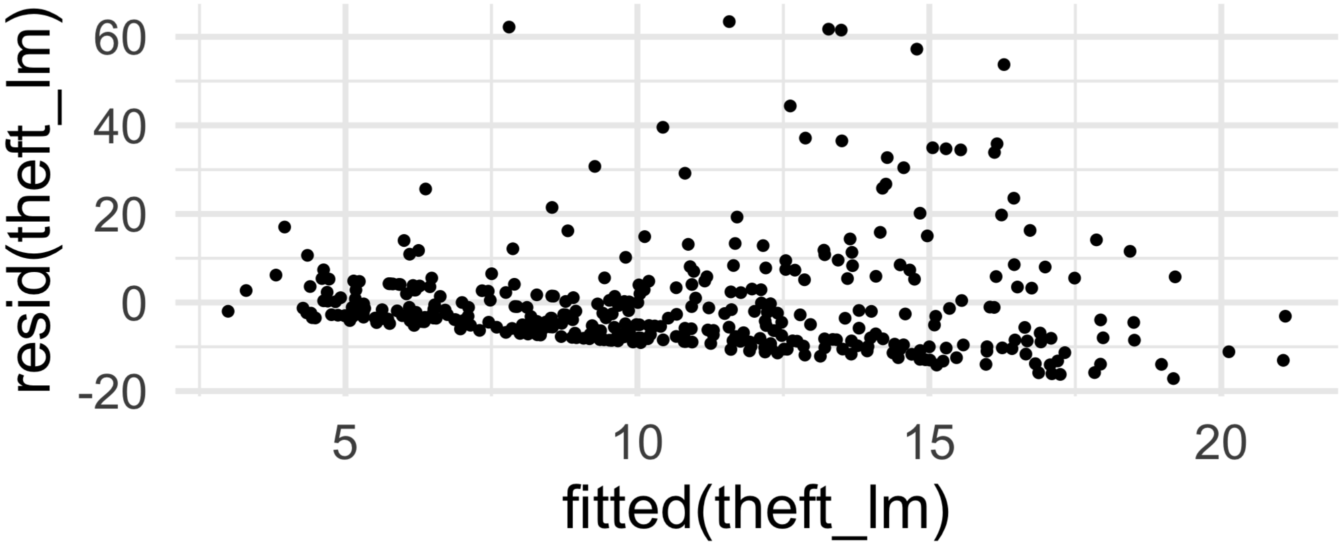
```
gf_point(Thefts ~ fitted(theft_lm),
         data = sscrime) |>
  gf_abline(intercept = 0, slope = 1)
```



What would this graph look like if `(R^2)` *were 1?*

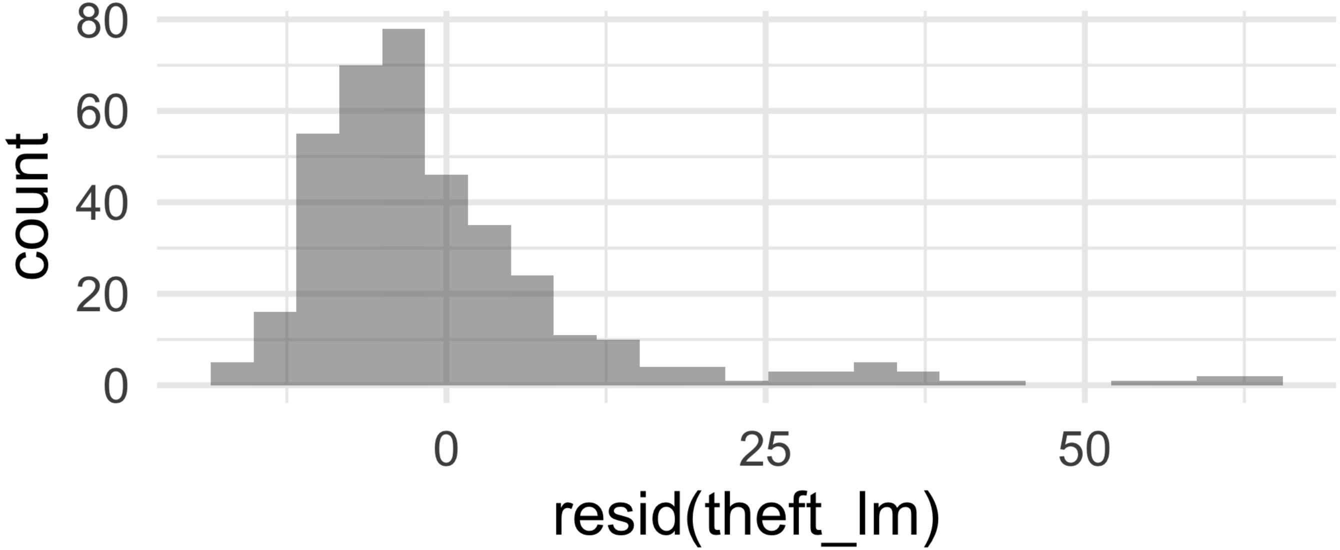
Assessment

```
gf_point(resid(theft_lm) ~ fitted(theft_lm))
```



Assessment

```
gf_histogram(~resid(theft_lm))
```



Problems

Solution: Count Regression

Adjust regression equation so that the model *expects* count data as the response

Which distribution(s) may come in handy?

Poisson Regression

(The math)

Poisson Regression - R

this is a GLM = "Generalized Linear Model"

```
theft_pois <- glm(Thefts ~ NEnrollment + Location +
                   TrainingHours + SecurityCameras,
                  data = sscrime,
                  family = poisson(link = 'log'))
```

Poisson Regression - R

Another way to do SAME thing - more extensible

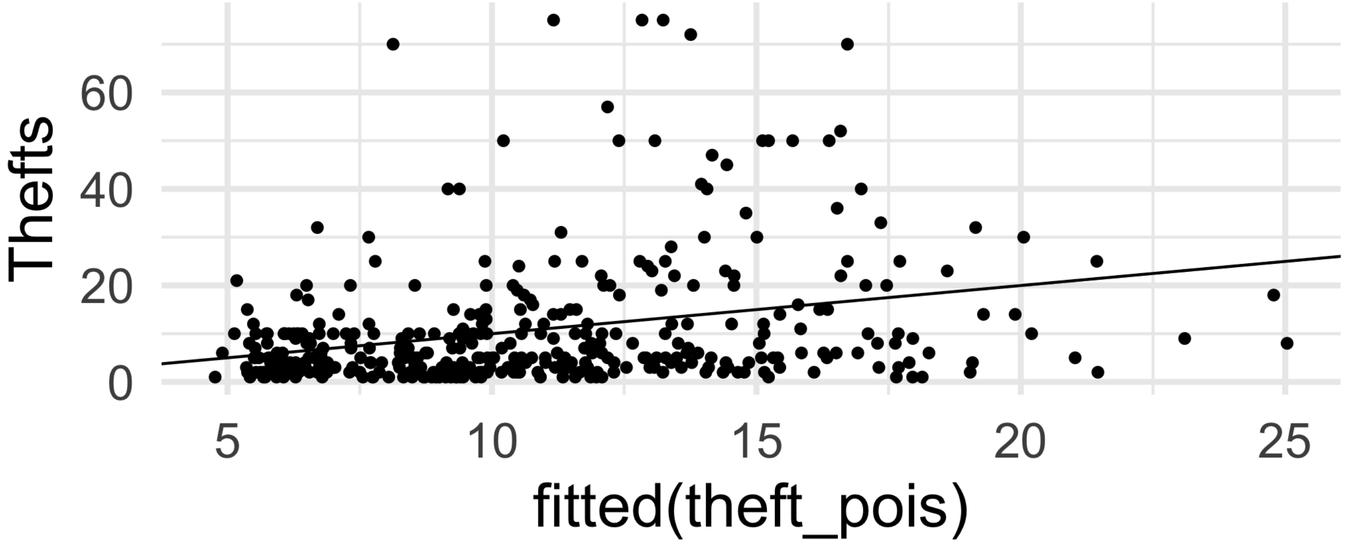
```
library(glmmTMB)
theft_pois <- glmmTMB(Thefts ~ NEnrollment + Location
+
                 TrainingHours + SecurityCameras,
               data = sscrime,
               family = poisson(link = 'log'))
```

```r
summary(theft_pois)
```

```
##  Family: poisson  ( log )
## Formula:
## Thefts ~ NEnrollment + Location + TrainingHours + SecurityCameras
## Data: sscrime
##
##      AIC      BIC   logLik deviance df.resid
##   5265.6   5293.2  -2625.8   5251.6      374
##
##
## Conditional model:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)            2.147e+00  4.544e-02   47.25  < 2e-16 ***
## NEnrollment            2.633e-04  9.646e-06   27.29  < 2e-16 ***
## LocationTown          -4.007e-01  5.630e-02   -7.12 1.11e-12 ***
## LocationUrban Fringe -8.364e-02  3.673e-02   -2.28   0.0228 *
## LocationRural         -5.145e-01  4.905e-02  -10.49  < 2e-16 ***
## TrainingHours         -2.480e-02  1.740e-02   -1.43   0.1540
## SecurityCamerasyes     2.019e-01  3.229e-02    6.25 4.01e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Response vs Fitted

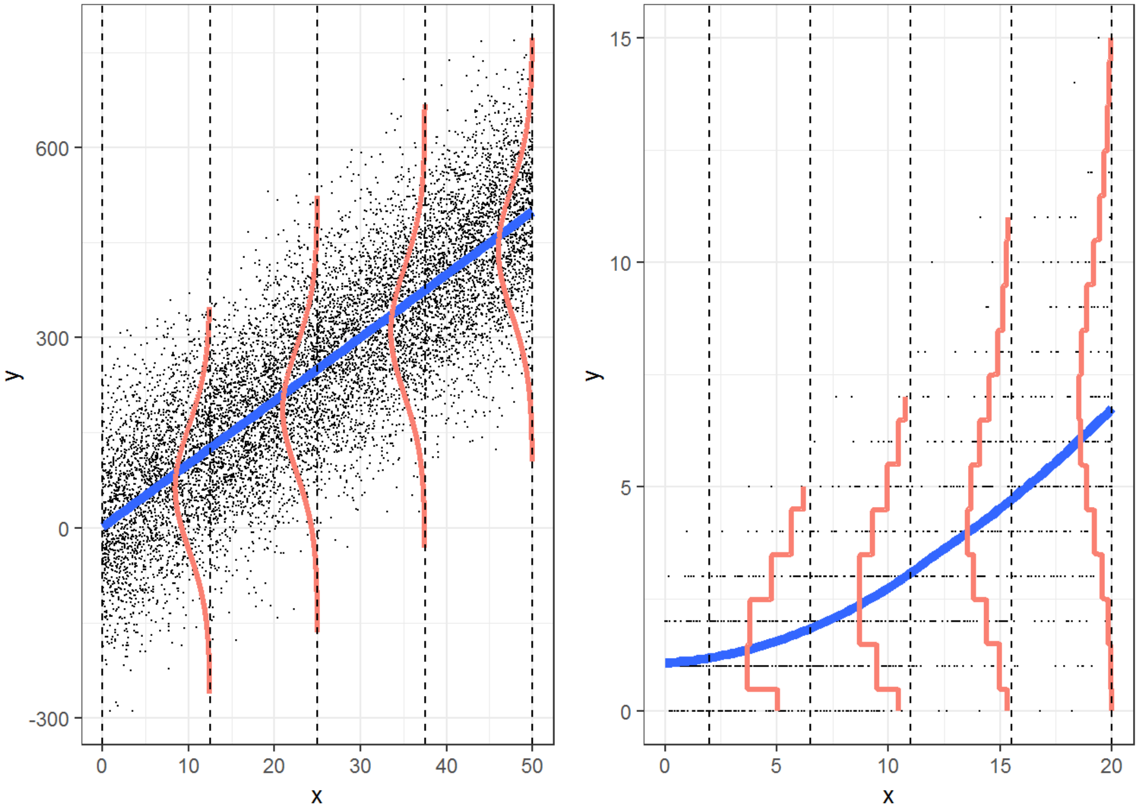
```
gf_point(Thefts ~ fitted(theft_pois),
         data = sscrime) |>
  gf_abline(intercept = 0, slope = 1)
```



Assessment for Poisson Regression (Conditions)

- Response (y) is *count data*
- (log)-Linearity: $log(\lambda_i)$ is a linear function of the covariates x_1, x_2, ... x_n
- Mean-variance relationship: Mean (of response) = Variance (of residuals)
- Independence (of residuals)
- **NO** Normal condition or other PDF residuals should follow

Poisson vs. Linear Model

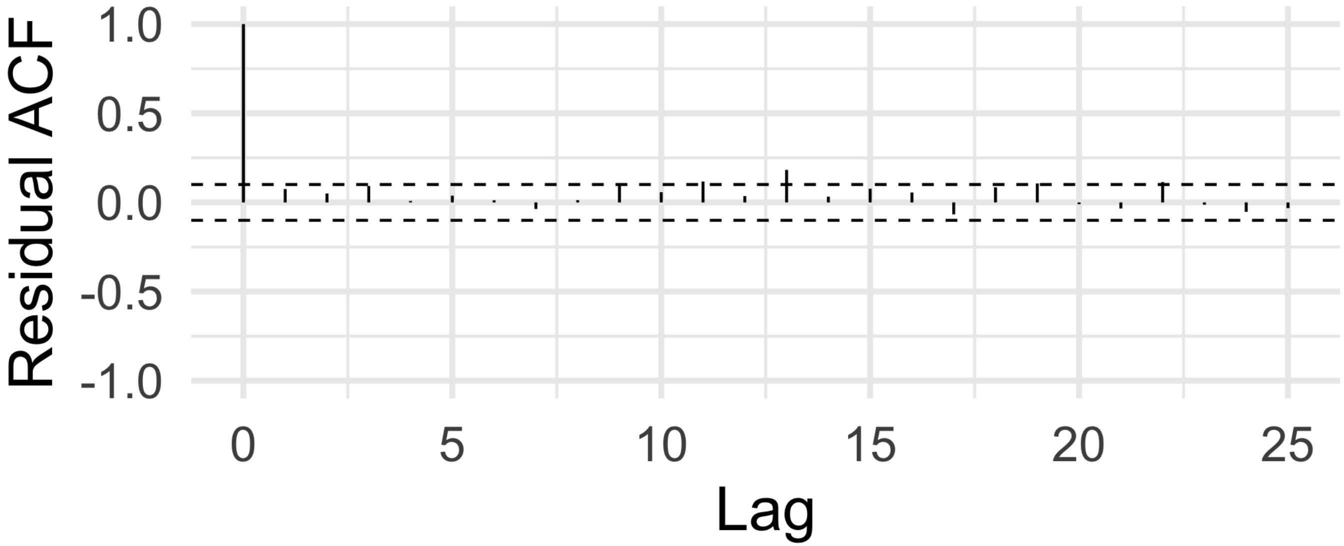


[Roback and Legler Section 4.2.2]()

Assessment

Independence - check it same as before

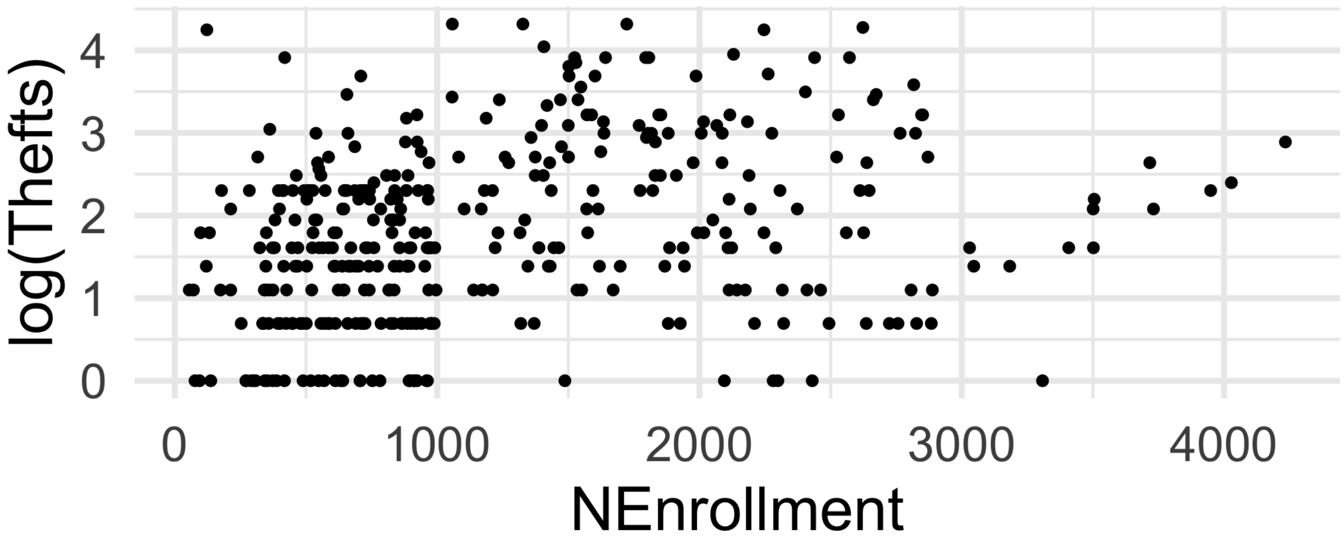
```
s245::gf_acf(~resid(theft_pois)) |> gf_lims(y =
c(-1,1))
```



Assessment

Linearity (for quantitative predictors)

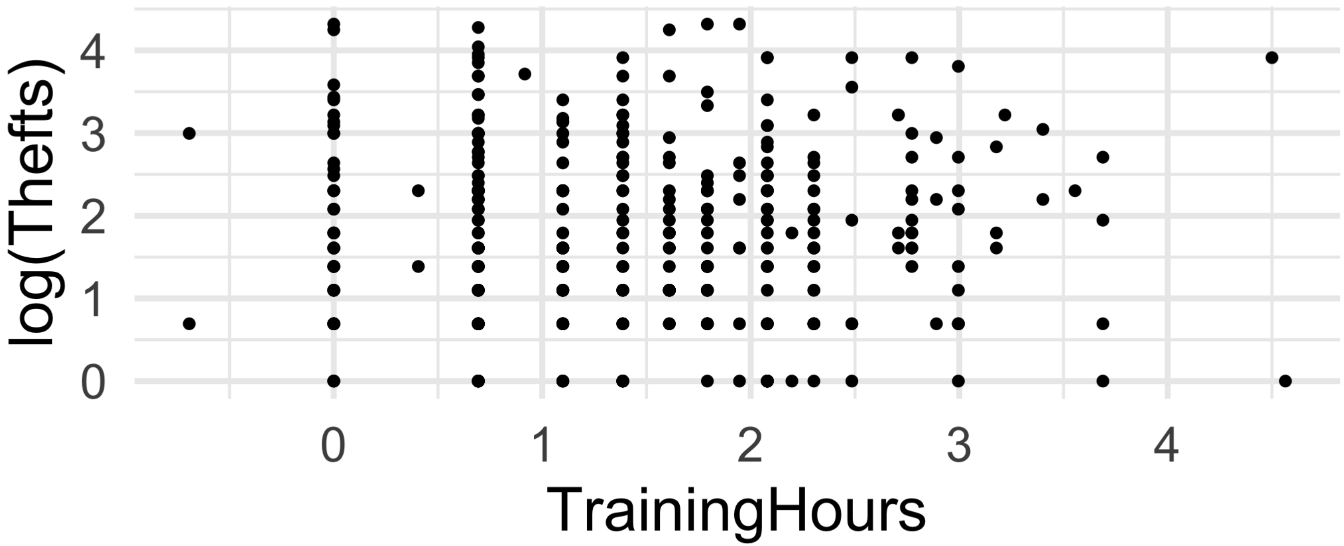
```
gf_point(log(Thefts) ~ NEnrollment,
         data =sscrime)
```



Assessment

Linearity (for quantitative predictors)

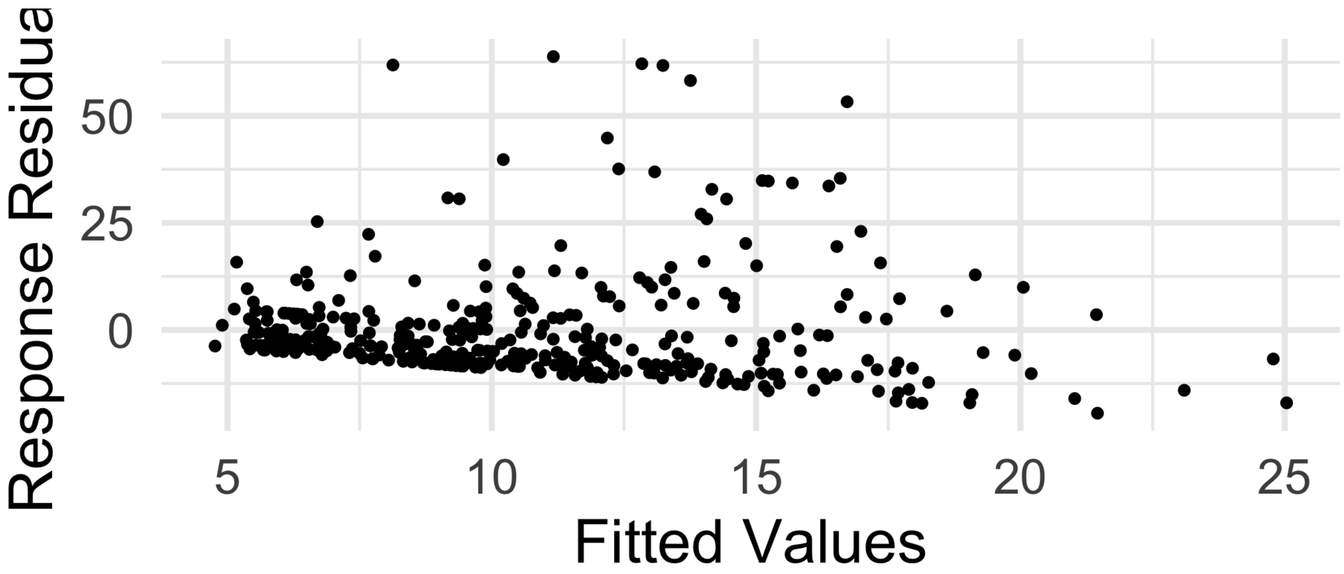
```
gf_point(log(Thefts) ~ TrainingHours,
         data =sscrime)
```



Assessment: Error Variance

Error Variance is equal to mean predicted count

```
gf_point(resid(theft_pois, type = 'response') ~
             fitted(theft_pois)) |>
  gf_labs(y = 'Response Residuals', x = 'Fitted
Values')
```

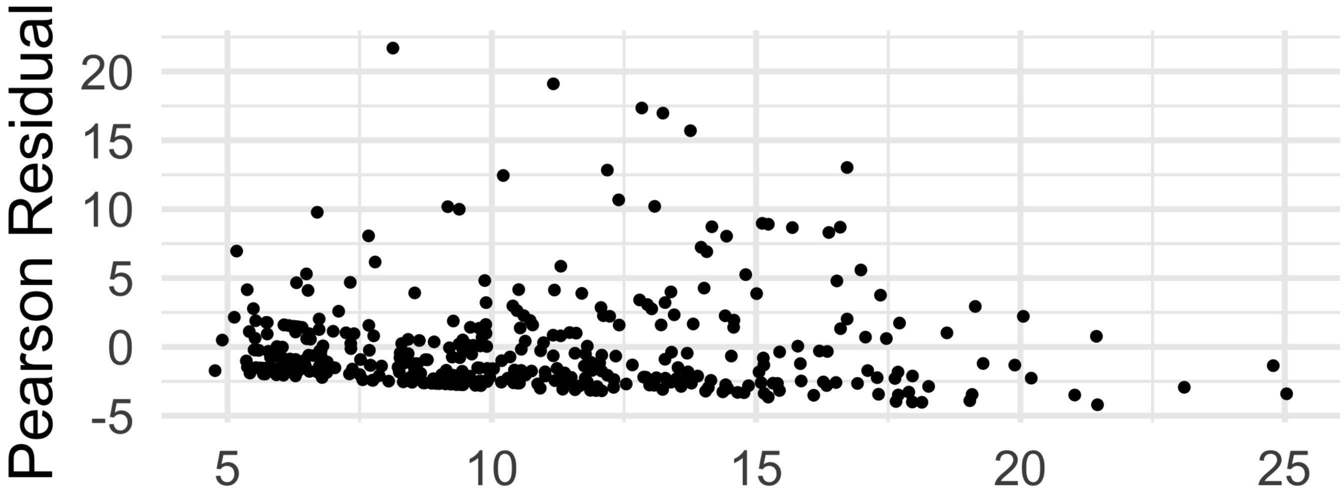


Assessment: Error Variance

DON'T REALLY USE THIS

Pearson residuals: scale each residual by expected variance

```
gf_point(resid(theft_pois, type = 'pearson') ~
           fitted(theft_pois)) |>
  gf_labs(y = 'Pearson Residuals', x = 'Fitted
Values')
```



Assessment: Error Variance

DON'T REALLY USE THIS

Bin residuals and plot mean, var for each bin

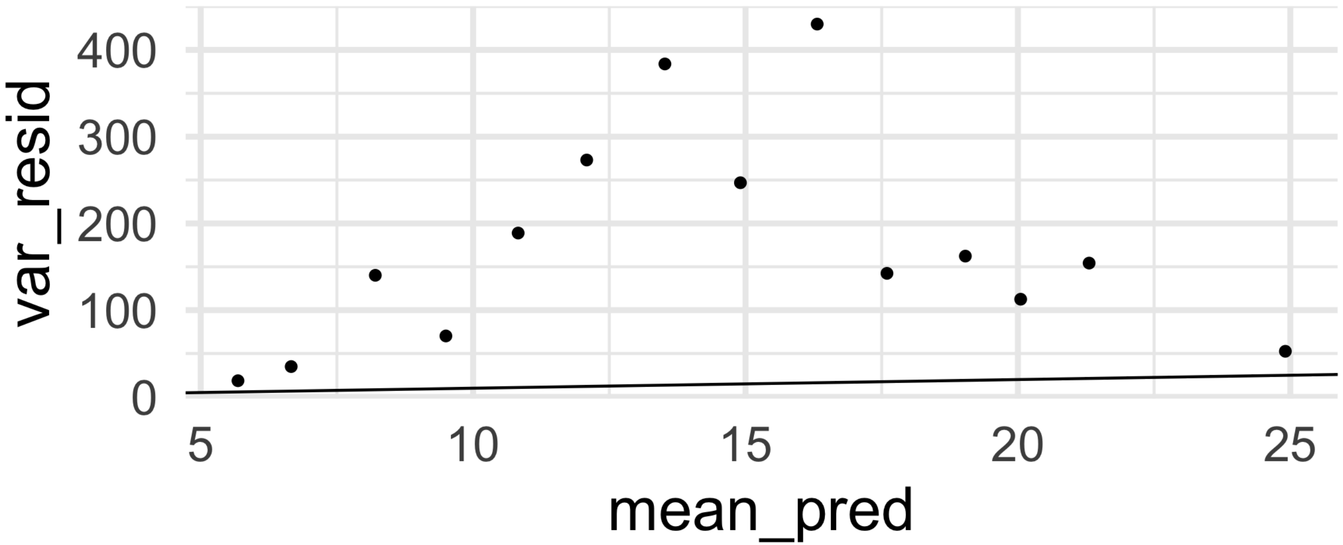
```
resid_mean_var <- sscrime |>
  mutate(preds = fitted(theft_pois),
         resids = resid(theft_pois, type =
'response'),
         pred_bins = cut(preds, 15)) |>
  group_by(pred_bins) |>
  summarize(mean_pred = mean(preds),
            var_resid = var(resids))
```

Assessment: Error Variance

DON'T REALLY USE THIS

Bin residuals and plot mean, var for each bin

```
gf_point(var_resid ~ mean_pred,
         data = resid_mean_var) |>
  gf_abline(intercept = 0, slope = 1)
```



Oh My Goodness Help

(and it will only get worse)

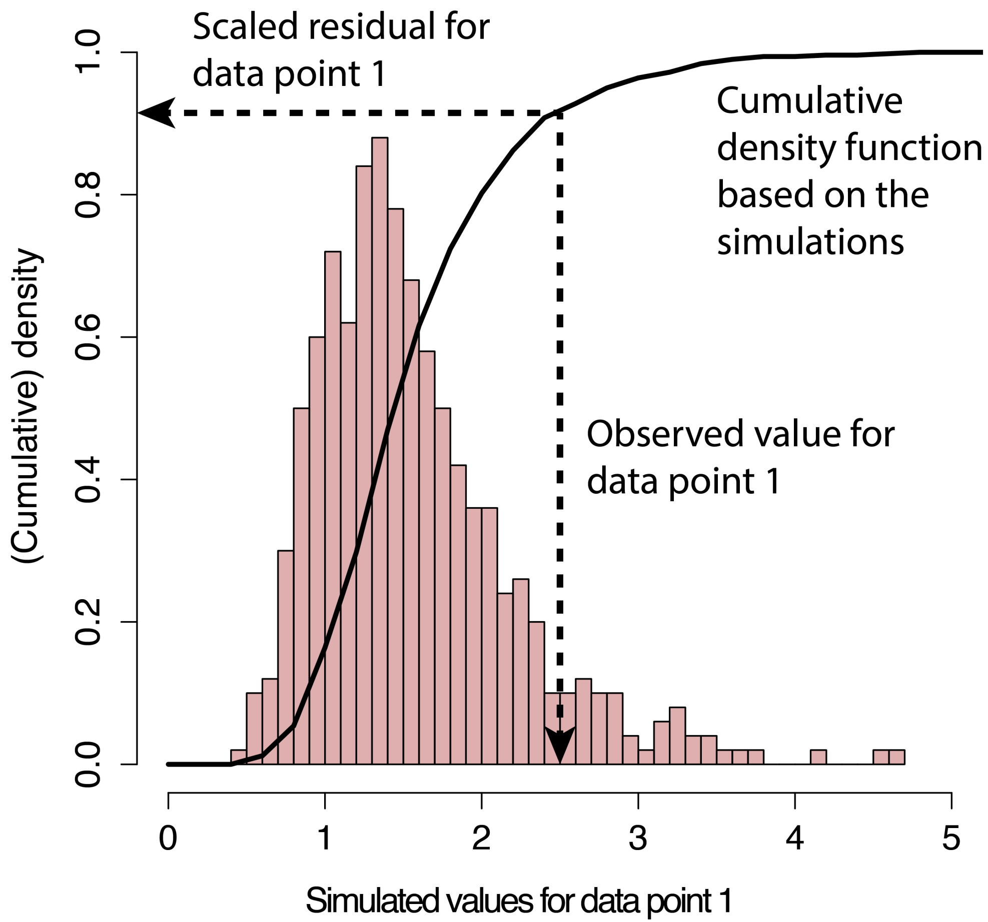
We need a better way to check mean-variance conditions henceforth.

- Simulate new data from fitted model
- Repeat to get expected distribution for each residual
- Standardize residuals on 0-1 scale (0 = all simulated resid are larger than real resid; 0.5 means half of simulated resids are larger than real resid).
- Standardized resids should be Uniform (0, 1)

Profiles in Statistics

Florian Hartig, University of Regensburg

Visually:

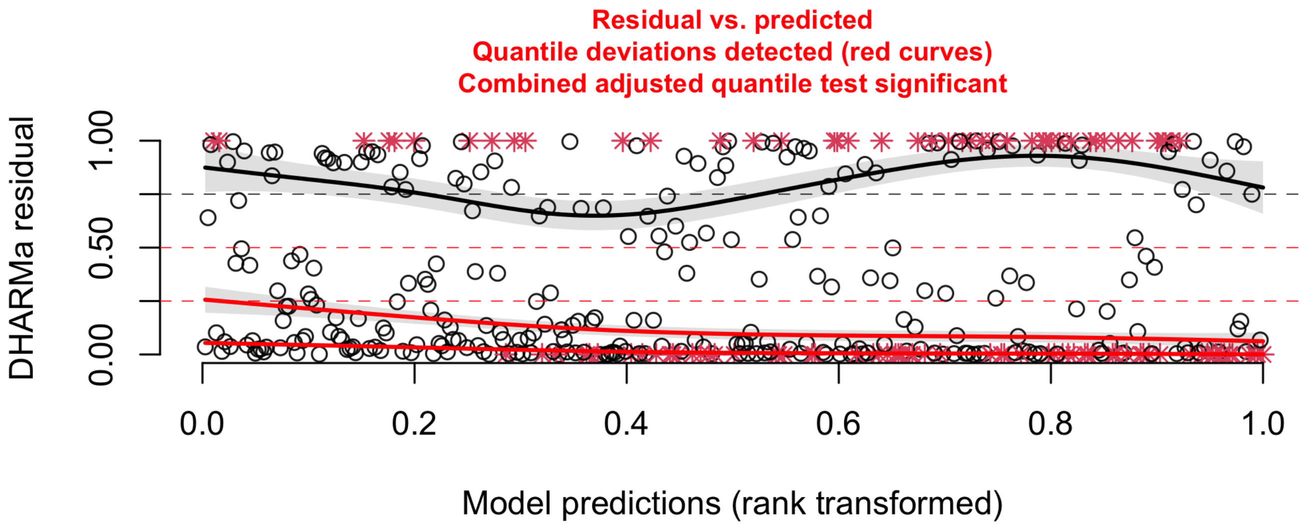


from https://cran.r-project.org/web/packages/DHARMa/vignettes/DHARMa.html

DHARMa Scaled Residuals

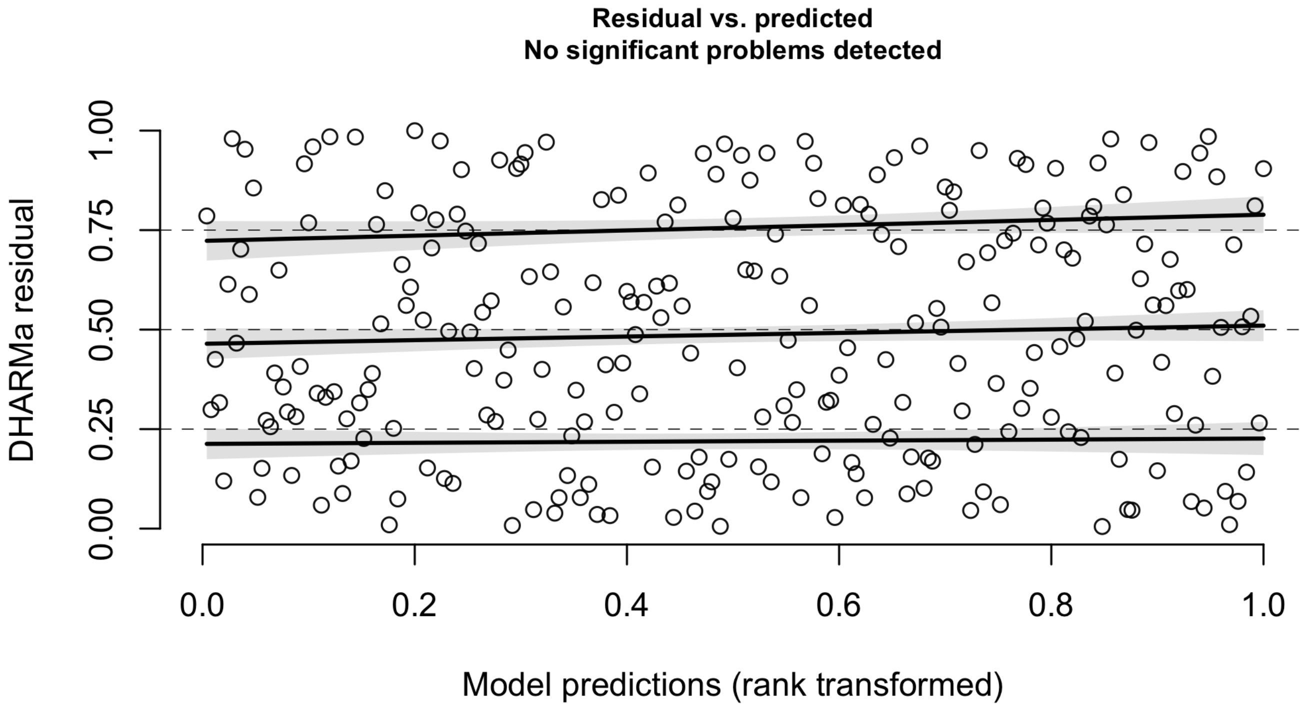
We can use for all models from now on! (even `lm()` ones!)

```
library(DHARMa)
pois_sim <- simulateResiduals(theft_pois)
plotResiduals(pois_sim)
```



DHARMa Scaled Residuals

How it *should* look if all is well: UNIFORM, trendless



Status: Limited Succeses

Poisson model seems *a bit* better for count data

- never predicts negative counts
- expects some "trumpet"
- New method to check resid vs fitted when not "normal, constant variance"

Status

Remaining Issues

- **Overdispersion common** (variance > mean: "super-trumpet"; Poisson's not good enough)
- Offsets? Counts per *what?* (Thefts per capita or per school?)